groupSum – sumaGrupos

Dado un arreglo de enteros, ¿Es posible escoger un grupo de algunos de los enteros, tal que la suma del grupo sea igual al entero objetivo (target)? Este es un problema clásico de backtracking. Una vez que entiendas la estrategia del backtracking en este problema, es posible usar el mismo patrón para muchos problemas para buscar un espacio de posibilidades. En lugar de buscar en el arreglo completo, nuestra convención es considerar la parte del arreglo a partir del índice (start) y continuando hasta el final del arreglo. El llamado puede especificar todo el arreglo simplemente pasando el inicio como cero. No se requieren ciclos ya que las llamadas avanzan en el arreglo.

groupSum6 – sumaGrupos6

Dado un arreglo de enteros, es posible escoger un grupo de algunos de los enteros, comenzando en el índice de inicio (start), tal que la suma de los elementos en el grupo sea igual al (entero dado) objetivo (target)? Sin embargo, con la siguiente restricción: todos los 6 debe ser incluidos (no requiere ciclos).

groupNoAdjt – grupoNoAdyt

Dado un arreglo de enteros, es posible escoger un grupo de algunos de los enteros, comenzando en el índice de inicio (start), tal que la suma de los elementos en el grupo sea igual al (entero dado) objetivo (target)? Sin embargo, con la siguiente restricción: si un valor en el arreglo es incluido en el grupo, el valor que sigue inmediatamente a este en el arreglo no puede ser incluido (no requiere ciclos).

groupSum5 – sumaGrupos5

Dado un arreglo de enteros, es posible escoger un grupo de algunos de los enteros, comenzando en el índice de inicio (start), tal que la suma de los elementos en el grupo sea igual al (entero dado) objetivo (target)? Sin embargo, con las siguientes restricciones: todos los múltiplos de 5 que estén en el arreglo deben ser incluidos en el grupo, y si el valor inmediatamente después de un múltiplo de 5 es 1, este valor no podrá ser incluido en el grupo (no requiere ciclos).

groupSumClump – sumaGruposGrumos

Dado un arreglo de enteros, es posible escoger un grupo de algunos de los enteros, comenzando en el índice de inicio (start), tal que la suma de los elementos en el grupo sea igual al (entero dado) objetivo (target)? Sin embargo, con la siguiente restricción: si hay números en el arreglo que son adyacentes y de valor idéntico, tenemos dos opciones: incluirlos a todos en el grupo, o no incluir a ninguno. Por ejemplo, con el arreglo {1, 2, 2, 2, 5, 2}, podemos incluir todos los 2s del medio o no incluir ninguno de ellos (puede usar un ciclo para encontrar la longitud de los grupos de valores adyacentes iguales).

splitArray – partirArreglo

Dado un arreglo de enteros, ¿Es posible dividir los enteros en dos grupos, tales que la suma de los dos grupos sean la misma? Cada entero debe estar en un grupo o en otro. Escribir un método recursivo auxiliar que tome cualquier argumento(s) que guste, y haga el llamado inicial a su función recursiva auxiliar desde splitArray() (no requiere ciclos).

splitOdd10 – partirImpar10

Dado un arreglo de enteros, ¿Es posible dividir los enteros en dos grupos, tales que la suma de un grupo es múltiplo de 10, y la suma del otro grupo es impar? Cada entero debe estar en un grupo o en otro. Escribir un método recursivo auxiliar que tome cualquier argumento(s) que guste, y haga el llamado inicial a su función recursiva auxiliar desde splitOdd10 () (no requiere ciclos).

split53 – partir53

Dado un arreglo de enteros, ¿Es posible dividir los enteros en dos grupos, tales que la suma de los dos grupos sean la misma? Pero esta vez, con las siguientes condiciones: todos los números que sean múltiplos de 5 deberán estar en un grupo, y todos los que sean múltiplos de 3 (y no múltiplos de 5) deberán estar en el otro. (no requiere ciclos).